E4934

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 05-233323
(43)Date of publication of application : 10.09.1993

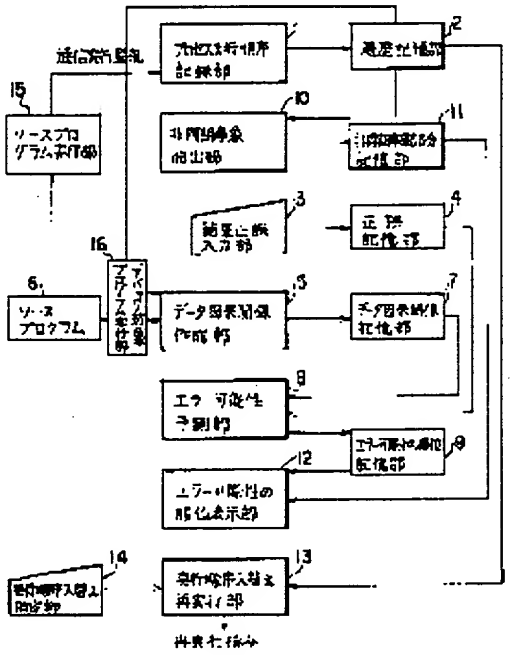
(51)Int.Cl. G06F 9/46
G06F 11/28

(21)Application number : 04-035353
(22)Date of filing : 21.02.1992

(71)Applicant : TOSHIBA CORP
(72)Inventor : KITAMURA ASAKO
HASEGAWA TETSUO
SEKI TOSHIBUMI

(54) DEBUG SUPPORTING DEVICE FOR PARALLEL PROGRAM

(57)Abstract:
PURPOSE: To efficiently perform debug interactively by combining required minimum events without performing a test by replacing the execution sequence of all the asynchronous parts of a debug target process.
CONSTITUTION: A data causal relation generating part 5 which generates data causal relation information from relation between the data of the debug target process and the execution sequence among processes based on program execution sequence information among the processes obtained by the execution of a source program 6 and information obtained by the execution of the debug target process by the source program 6, and a display part 12 which displays the possibility of a program error predicted based on the data causal relation generated at the data causal relation generating part 5 and the program execution sequence among the processes as debug supporting information are provided at a computer system which performs processing as performing message communication among plural processes by the execution of the source program 6.



LEGAL STATUS

[Date of request for examination] 14.01.1999
[Date of sending the examiner's decision of rejection]
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of rejection]
[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平5－233323

(43)公開日　平成５年(1993)９月10日

(51)Int.Cl.[5]	識別記号	庁内整理番号	Ｆ Ｉ	技術表示箇所
Ｇ０６Ｆ　9/46	３３０　Ｃ	8120－5B		
11/28	Ａ	9290－5B		

審査請求　未請求　請求項の数３(全 11 頁)

(21)出願番号　特願平4－35353

(22)出願日　平成４年(1992)２月21日

(71)出願人　000003078
株式会社東芝
神奈川県川崎市幸区堀川町72番地

(72)発明者　北村　麻子
神奈川県川崎市幸区柳町70番地　株式会社東芝柳町工場内

(72)発明者　長谷川　哲夫
神奈川県川崎市幸区柳町70番地　株式会社東芝柳町工場内

(72)発明者　關　俊文
神奈川県川崎市幸区柳町70番地　株式会社東芝柳町工場内

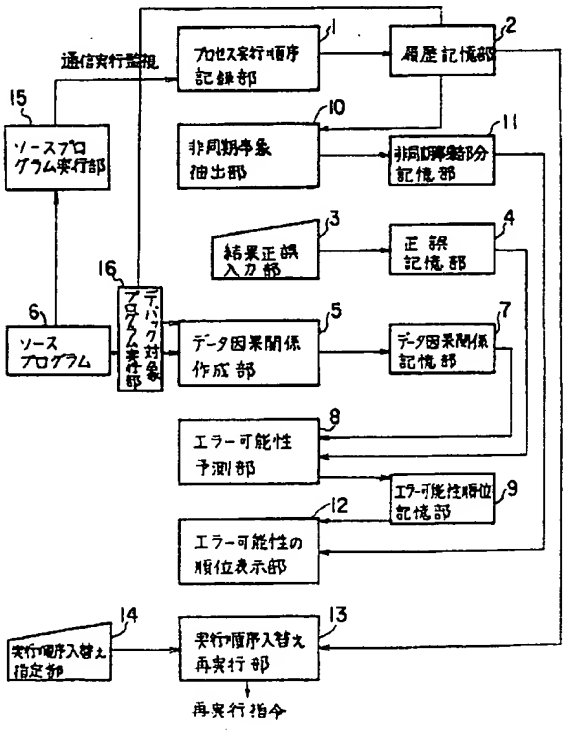
(74)代理人　弁理士　鈴江　武彦

(54)【発明の名称】　並行プログラムのデバッグ支援装置

(57)【要約】

【目的】デバッグ対象プロセスの非同期部分の全てに対して実行順序を入替えてテストすることなく、必要最小限の非同期事象の組合せでデバッグを対話的に効率良く行うことにある。

【構成】ソースプログラム６の実行により複数のプロセス間でメッセージ通信をしながら処理を進める計算機システムにおいて、ソースプログラム６の実行により得られるプロセス間のプログラム実行順序情報とソースプログラム６によるデバッグ対象プロセスの実行により得られる情報とに基づき前記デバッグ対象プロセスのデータと前記プロセス間の実行順序との関係からデータ因果関係情報を作成するデータ因果関係作成部５と、このデータ因果関係作成部５で作成されたデータ因果関係とプロセス間のプログラム実行順序に基づいて予測されるプログラムエラーの可能性をデバッグ支援情報として表示する表示部１２とを備える。



【特許請求の範囲】
【請求項１】　ソースプログラムの実行により複数のプロセス間でメッセージ通信をしながら処理を進める計算機システムにおいて、前記ソースプログラムの実行により得られるプロセス間のプログラム実行順序情報を用いて前記ソースプログラムによるデバッグ対象プロセスの実行により得られる情報に基づき前記デバッグ対象プロセスのデータの定義およびデータ間の参照関係からデータの因果関係情報を作成するデータ因果関係作成手段と、このデータ因果関係作成手段で作成されたデータ因果関係に前記ソースプログラムの実行により得られた結果の正誤をもとに予測したデータの引数として受取る部分でのプログラムエラーの可能性をデバッグ支援情報として表示する表示手段とを備えたことを特徴とする並行プログラムのデバッグ支援装置。
【請求項２】　表示手段に表示されるデバッグ支援情報は、データ因果関係情報にプロセス間のプログラム実行順序から抽出された実行順序の入替え可能な非同期事象部分を合わせて表示可能な情報としたことを特徴とする請求項１に記載の並行プログラムのデバッグ支援装置。
【請求項３】　ソースプログラムの実行により複数のプロセス間でメッセージ通信をしながら処理を進める計算機システムにおいて、前記ソースプログラムの実行により得られるプロセス間のプログラム実行順序情報と前記ソースプログラムによるデバッグ対象プロセスの実行により得られる情報とに基づき前記デバッグ対象プロセスのデータと前記プロセス間のプログラム実行順序との関係からデータ因果関係情報を作成するデータ因果関係作成手段と、このデータ因果関係作成手段で作成されたデータ因果関係に前記ソースプログラムの実行により得られた結果をもとに予測したデータを引数として受取る部分でのプログラムエラーの可能性とプロセス間のプログラム実行順序に基づいて予測される実行順序の入替可能な非同期事象部分を併せてデバッグ支援情報として表示する表示手段と、この表示手段に表示された表示内容に従って前記非同期事象部分に対する実行順序の入替えが指定されると前記プロセス間のプログラム実行順序情報をもとに前記デバッグ対象プロセスの非同期事象に対する実行順序を入替える再実行指令を出力する実行順序入替え再実行手段とを備えたことを特徴とする並行プログラムのデバッグ支援装置。
【発明の詳細な説明】
【０００１】
【産業上の利用分野】本発明は複数のプロセス間でメッセージ通信をしながら処理を進める計算機システムにおいて、デバッグの対象となるプロセス（以下デバッグ対象プロセスと呼ぶ）に存する実行順序の入替可能な非同期事象部分で生じるプログラムエラーを分析、検出するための並行プログラムのデバッグ支援装置に関する。
【０００２】
【従来の技術】複数のプロセス間でメッセージ通信をしながらプログラム処理を進める計算機システムにおいて、プロセス中にメッセージに対する同期の取り忘れや、意図的に同期を取っていない部分（実行順序の入替可能な非同期事象部分：ここでは単に非同期事象部分と呼ぶ）があると、他のプロセスからのメッセージの到着する順番が変わった場合には、ユーザの意図しないプロセス間の通信によるプログラムエラーが発生することがある。従って、このプログラムエラーを検出するためには、予めプログラムエラーとなる非同期事象部分がどこであるかを検索しておく必要がある。
【０００３】従来、かかるプログラムエラーとなる非同期事象部分を検索するには、まずソースプログラムの実行により各プロセス間のプログラム実行順序を取込んで、デバッグ対象プロセス中に存する非同期事象部分を抽出し、次にデバッグ対象プロセスの非同期事象部分の全ての組合わせで実行順序を入替えてテストすることによりプログラムエラーとなる非同期事象部分を検索するようにしていた。
【０００４】
【発明が解決しようとする課題】しかし、このような検索手法では、全ての非同期事象部分に対して実行順序を入替えて再実行テストをしなければならず、しかもその組合わせ数も多数存在するため、デバッグに膨大な時間と手間がかかるという問題がある。
【０００５】本発明は、デバッグ対象プロセスの非同期事象部分の全てに対して実行順序を入替えてテストすることなく、必要最小限の非同期事象部分の組合せでデバッグを対話的に効率良く行うことができる並行プログラムのデバッグ支援装置を提供することを目的とする。
【０００６】
【課題を解決するための手段】本発明は上記の目的を達成するため、ソースプログラムの実行により複数のプロセス間でメッセージ通信をしながら処理を進める計算機システムにおいて、前記ソースプログラムの実行により得られるプロセス間のプログラム実行順序情報と前記ソースプログラムによるデバッグ対象プロセスの実行により得られる情報とに基づき前記デバッグ対象プロセスのデータと前記プロセス間のプログラム実行順序との関係からデータの因果関係情報を作成するデータ因果関係作成手段と、このデータ因果関係作成手段で作成されたデータ因果関係とプロセス間のプログラム実行順序に基づいて予測される実行順序の入替可能な非同期事象部分でのプログラムエラーの可能性をデバッグ支援情報として表示する表示手段とを備えたものである。
【０００７】また、ソースプログラムの実行により複数のプロセス間でメッセージ通信をしながら処理を進める計算機システムにおいて、前記ソースプログラムの実行により得られるプロセス間のプログラム実行順序情報と前記ソースプログラムによるデバッグ対象プロセスの実

行により得られる情報とに基づき前記デバッグ対象プロセスのデータと前記プロセス間のプログラム実行順序との関係からデータ因果関係情報を作成するデータ因果関係作成手段と、このデータ因果関係作成手段で作成されたデータ因果関係とプロセス間のプログラム実行順序に基づいて予測される実行順序の入替可能な非同期事象部分でのプログラムエラーの可能性をデバッグ支援情報として表示する表示手段と、この表示手段に表示された表示内容に従って前記非同期事象部分に対する実行順序の入替えが指定されると前記プロセス間のプログラム実行順序情報をもとに前記デバッグ対象プロセスの非同期事象に対する実行順序を入替える再実行指令を出力する実行順序入替え再実行手段とを備えたものである。

【０００８】

【作用】このような構成の並行プログラムのデバッグ支援装置にあっては、データ因果関係作成手段により、ソースプログラムの実行により得られるプロセス間のプログラム実行順序情報とソースプログラムによるデバッグ対象プロセスの実行により得られる情報とに基づきデバッグ対象プロセスのデータとプロセス間の実行順序との関係からデータ因果関係情報が作成され、このデータ因果関係とプロセス間のプログラム実行順序に基づいて予測される実行順序の入替可能な非同期事象部分でのプログラムエラーの可能性がデバッグ支援情報として表示手段に表示されるので、プログラムエラーとなる実行順序の入替えが可能な非同期事象部分の検索が容易に行うことができる。

【０００９】また、実行順序入替え再実行手段により表示手段に表示された表示内容に従ってプログラムエラーの可能性の高いデバッグ対象プロセスの非同期事象に対する入替えを行って再実行されるので、デバッグ対象プロセスの非同期部分の全てに対して実行順序を入替えてテストすることなく、必要最小限の非同期事象の組合せでデバッグ支援を対話的に効率良く行うことが可能となり、デバッグに要する時間および手間を大幅に削減することができる。

【００１０】

【実施例】以下本発明の一実施例を図面を参照して説明する。

【００１１】図１は本発明による並行プログラムのデバッグ支援装置の構成例を示すブロック図である。図１において、１はソースプログラム６を実行するソースプログラム実行部１５によりプロセス間で実行されるメッセージ通信を取込んでプログラム実行順序を記録するプログラム実行順序記録部、２はこのプログラム実行順序が通信履歴として書込まれる履歴記憶部、３はこのソースプログラム実行により得られたデバッグ対象プロセスの出力値が正しいか誤っているかの情報を入力する正誤入力部、４はこの正誤入力部３より出力される正誤情報を記憶する正誤記憶部である。

【００１２】また、５はデバッグ対象プロセス実行部１６により取込まれたソースプログラム６からの情報と履歴記憶部２に記憶されたプロセス間の実行順序の履歴に基づいてデータの因果関係を解析し、デバッグ対象プロセスのデータ間の参照と定義の関係を含むデータ因果関係情報を作成するデータ因果関係作成部、７はこのデータ因果関係作成部５で作成されたデータ因果関係情報を記憶するデータ因果関係記憶部、８はこのデータ因果関係記憶部７に記憶されたデバッグ対象プロセスのデータ因果関係情報と正誤記憶部４に記憶された正誤情報を用いてプログラムエラーの可能性を計算し、プログラムエラーが含まれる可能性の高さを順位付けするエラー可能性予測部で、このエラー可能性予測部８で予測されたプログラムエラー予測性の順位はエラー可能性順位記憶部９に記憶される。

【００１３】一方、１０は履歴記憶部２に記憶されているプロセス間のプログラム実行順序の情報を取込んで同期事象であるか非同期事象であるかを判断して非同期事象部分を抽出する非同期事象抽出部で、この非同期事象抽出部１０により抽出された非同期事象部分は非同期事象部分記憶部１１に記憶される。また、１２はエラー可能性順位記憶部９に記憶されているエラー可能性の順位と非同期事象部分記憶部１１に記憶されている非同期事象部分を表示する表示部である。さらに、１３は実行順序の入替え通信指定部１４より表示部１２に表示された表示内容に従ってプロセス間の実行順序を入替えるメッセージが入力されると共に、履歴記憶部２より実行順序を取込んでその実行順序の入替えの再実行指令を出力する実行順序入替え再実行部である。

【００１４】次に上記のよう構成された並行プログラムのデバッグ支援装置の作用について述べる。いま、例えば図２に示すような３つのプロセスＸ，Ｙ，Ｚからなる並行プログラムがあり、デバッグ対象プロセスＸが求める結果がＫ１，Ｋ２，Ｋ３の値であるとする。また、今回のプロセス間のプログラム実行順序が、図３のように行われたとする。このときのデバッグ装置の処理の流れを図４および図５に示し、図４におけるステップ104 の詳細を図６に、図４におけるステップ105 の詳細を図７乃至図９に示す。

【００１５】まず、図４において、ステップ101 により実行が開始されると、ソースプログラム実行部１５によりソースプログラム６の情報に基いて各プロセスを実行し、そのときのプロセス間のプログラム実行順序の監視により、そのプロセス間のプログラム実行順序がプロセス実行順序記録部１に取込まれる。このプロセス実行順序記録部１に取込まれたプログラム実行順序は、履歴記憶部２に履歴データとして格納される。図１０は履歴記憶部２に格納された履歴データを示すものである。次にステップ102 によってデバッグ対象プロセスを決める。図２の並行プログラムでは、出力結果がＫ１，Ｋ２，Ｋ

３の値なので、ここではデバッグ対象プロセスをプロセスＸとする。次にステップ103 ではプロセス間のプログラム実行により得られるプログラム実行順序の出力結果から判断されるプロセスＸのＫ１，Ｋ２，Ｋ３の値の正誤情報を正誤入力手段３より正誤記憶部４に入力する。ここでは、説明の理解を容易にするためＫの中身は図１１に示すようにＫ１，Ｋ２の値が誤りで、Ｋ３の値が正しいものとして扱う。

【００１６】次にステップ104 では、デバッグ対象プロセス実行部１６の実行によりデータ因果関係作成部５にソースプログラム６からの情報および履歴記憶部２に記憶されているプログラム実行順序の履歴が取込まれると、このデータ因果関係作成部５ではこれらの情報に基づいてデータの因果関係を解析し、デバッグ対象プロセスのデータ間の参照と定義の関係を含むデータ因果関係の作成を行い、その情報をデータ因果関係記憶部７に格納する。ここでは、図２に示すような並行プログラムの場合を例としているので、図１２に示すようなデータ因果関係が作成される。

【００１７】ここで、図６に示すフローチャートによりデータ因果関係の作成手順について詳細に述べる。図６において、ステップ401 ではデバッグ対象プロセス実行部１６によりプロセスＸのみの再実行を行い、履歴記憶部２に格納されているプロセス間のプログラム実行順序の履歴とソースプログラム６からの情報をデータ因果関係作成部５に取込んで、パスを決める。この再実行により図２のＸ２行の分岐文では（２）が、Ｘ５行の分岐文では（４）がとられたとする。図２のプロセスＸの中では、Ｘ１，Ｘ３，Ｘ４，Ｘ６，Ｘ８行でＡ，Ｂ，Ｃの値を受信し、それらをもとにＶ，Ｗ，Ｋ１，Ｋ２，Ｋ３を計算して出力しているので、図６のステップ402 と403 によりＡ，Ｂ，Ｃの値が入力変数ノードとして、Ｖ，Ｗの値が中間ノードとして、またＫ１，Ｋ２，Ｋ３の値が出力ノードとしてデータ因果関係記憶部７に記憶される。そして、ステップ404 によりデータ因果関係記憶部７にあるノードをつないでいく。これを図２のプロセスＸについてみると、Ｘ５行の分岐文では（２）を通り、ＡからＫ１の値に影響を与えているので、ＡからＫ１をつなぐ。Ｘ５行の分岐文では（４）を通るため、Ａ，ＢがＶに影響を与える。そこで、ＡからＶとＢからＶをつなぐ。Ｘ７行ではＷの決定にＢが影響を与えているので、ＢからＷをつなぐ。Ｘ９行ではＶからＫ２、Ｘ１０ではＷとＣからＫ３をつなぐ。このように各ノードどうしをつないでいくことにより、図１２のデータ因果関係図が作成できる。

【００１８】再び図４に戻り、ステップ105 ではいままでの処理で得た正誤記憶部４にある正誤表（図１１）とデータ因果関係記憶部７にあるデータ因果関係図（図１２）を用いてエラー可能性予測部８によってどの受信動作によるメッセージの入力にエラーが含まれているかどうかを可能性の高さで順位づけ、エラー可能性順位記憶部９に図１３に示すようなエラー順位表を格納する。

【００１９】ここで、エラー可能性予測部８でのエラー可能性予測と順位付けの処理の流れについて図７乃至図９により詳細に述べる。まず、図７のステップ501 によりエラー順位表の変数名にデータ因果関係図のノード名を登録し、ステップ502 で図１１の正誤表から１つの変数の正誤を読み出し、ステップ503 で正しいか否かを判定し、正しければステップ504 にて点数を＋１に、正しくなければステップ505 にて点数を－１として出力変数Ｋ１，Ｋ２，Ｋ３に点数－１，－１，１が代入され、ステップ506 にて遷移数が確定するとフラグが１にセットされる。そして、ステップ507 にてすべての出力変数について終了したか否かが判定され、終了したことが判定されるとステップ508 に進む。次にこのステップ508 により中間変数Ｖが選ばれると、図８のステップ509 により１個前のノードに確定フラグが立っているか否かを判定する。すなわち、図１２のデータ因果関係図を参照すると、ＶからＫ２に線が引かれているので、これを辿り１つ前のノードであるＫ２に到達し、Ｋ２は確定フラグが立っているので、ステップ510 によりこの点数と遷移数をＶの点数と遷移数に加算する。これにより、Ｖは－１，１となる。その後辿ったことのない線は存在しないので、ステップ514 によりＶに確定フラグが立つ。

【００２０】次に図７のステップ508 によりＢが選ばれると、ＢからＶへの線を辿り、Ｖに確定フラグが立っていることから、Ｂの点数と遷移数に－１，１が加算される。次に再び図８のステップ509 によりＢからＷへの線を辿ってＷに到達するが、Ｗに確定フラグが立っていないので、ステップ511 によりＢは、点数、遷移数とも－１，１のままに次のＷの値の計算にいく。同様にして、Ｗの点数、遷移数が１，１と確定した後、再びＢのノードがステップ508 で選ばれたとき、ＢからＷを辿り、Ｂの－１，１の値にＷの１，１が加算され、最終的にＢの値は０，２になって確定する。

【００２１】このようにして、Ａ，Ｃの点数と遷移数も図７のステップ508 から図８のステップ515 によって確定する。すべての点数が確定した後は、ステップ516 により入力変数Ａ，Ｂ，Ｃに関しての順位づけが行われる。Ａに関しては、点数－２、遷移数２なので絶対値が等しく点数が負なので、図９のステップ520 によりエラーの可能性は最上位となる。Ｃはステップ521 により絶対値が等しく、点数が正なので最下位となる。このようにして、エラーの可能性は高い順にＡ，Ｂ，Ｃとなり、最終的に図１３に示すようなエラー順位表ができあがる。

【００２２】次に再び図４に戻り、エラー可能性順位記憶部９にあるエラー順位表の表示とともに、その受信による入力が同期事象であるか、非同期事象であるかを合せて表示するために、ステップ106 で非同期事象抽出部

１０を用いて履歴記憶部２にある図１０の履歴を解析する。これを見ると、図３のプロセス間のプログラム実行順序が分かり、図３の［１］のプロセスＹからプロセスＸのデータＡの通信と［２］のプロセスＺからプロセスＸへのデータＡの通信の間には同期がとられていないため、この２つのプログラム実行順序が確定できない非同期事象部分であることが分かる。同様にプロセスＸあてのデータＢの通信［３］と［４］の場合も非同期事象である。これにより、ステップ107 では受信動作にエラーが含まれている可能性の高さを順位で示し、その受信動作が同期事象であるか、非同期事象であるかを示す。

【００２３】ステップ107 の表示に従ってユーザが非同期事象部分で、かつエラーの可能性の一番高いデータＡを選ぶと、図５のステップ110 によって実行順序入替え再実行部１３によりメッセージの到着順を図３における［１］［２］の順序ではなく、［２］［１］の順序に換えて再実行する。これにより、結果が正しい値に変わったときには、この非同期事象部分がエラーを引き起こした原因と考えられ、この部分に同期事象をいれるべきであることがわかる。結果に改善が見られない場合は、またエラーの可能性の順位に従って再実行をしてデバッグ支援を進める。

【００２４】このように本実施例では、３つのプロセスＸ，Ｙ，Ｚからなる並行プログラムによりメッセージ通信をしながら処理を進める計算機システムにおいて、予めソースプログラムの実行によりプロセス間のプログラム実行順序をプログラム実行順序記録部１に取込んでこれを履歴記憶部２に格納し、次にデバッグ対象プロセスの実行により履歴記憶部２に格納された実行順序の履歴とソースプログラム６からの情報をデータ因果関係作成部５に取込み、これらの情報に基づいてデータの因果関係を解析し、デバッグ対象プロセスのデータ間の参照と定義の関係を含むデータ因果関係を作成するようにしたので、このデータの因果関係からデバッグ対象プロセスに存するプログラムエラーにつながる非同期事象部分の解析を支援することが可能となる。

【００２５】また上記構成に加えて、上記履歴記憶部２に格納されたプロセス間のプログラム実行順序の履歴を非同期事象抽出部１０に取込んでプログラム実行順序が同期事象であるか非同期事象であるかを判断して非同期事象部分を抽出すると共に、デバッグ対象プロセスのプログラム実行により得られるプロセスＸの出力値Ｋ1，Ｋ2，Ｋ3 の結果に対して正しいか誤っているかの情報を正誤入力部３により入力してこれを一旦正誤記憶部４に格納しておき、この出力結果の正誤情報をエラー可能性予測部８に上記データ因果関係作成部５で作成されたデータの因果関係と共に入力して各メッセージ通信によって引き渡される値にエラーが含まれている可能性の強さを予測し、このエラーが含まれている可能性の強さを非同期事象抽出部１０により抽出された非同期事象部分と共に表示部１２に表示するようにしたので、必要最小限の非同期事象部分に対する実行順序の入替えで非同期事象部分のデバッグ支援を対話的に効率よく行うことができる。

【００２６】さらに、実行順序入替え指定部１４より実行順序入替え再実行部１３に表示部１２に表示された表示内容に従って入替えるメッセージを指定することにより履歴記憶部２よりプロセス間のプログラム実行順序の履歴を取込んで実行順序入替えの再実行指令が出力されるので、全ての非同期事象部分を組合わてテストする必要がなく、デバッグに要する時間および手間を大幅に短縮することができる。

【００２７】

【発明の効果】以上の述べたように本発明によれば、デバッグ対象プロセスの非同期事象部分の全てに対して実行順序を入替えてテストすることなく、必要最小限の非同期事象部分の組合せでデバッグを対話的に効率良く行うことができる並行プログラムのデバッグ支援装置を提供できる。

【図面の簡単な説明】

【図１】本発明による並行プログラムのデバッグ支援装置の一実施例を示すブロック構成図。

【図２】３つのプロセスからなる並行プログラムの一例を示す図。

【図３】同実施例におけるプロセス間の実行順序を示す図。

【図４】同実施例の作用を説明するためのフローチャートを示す図。

【図５】図４に続くフローチャートを示す図。

【図６】図４のデータ因果関係作成ステップの詳細を示すフローチャートを示す図。

【図７】図４の非同期事象抽出ステップの詳細を示すフローチャートを示す図。

【図８】図７に続くフローチャートを示す図。

【図９】図８に続くフローチャートを示す図。

【図１０】図１の履歴記憶部に格納されたプロセス間の実行順序の履歴の一例を示す図。

【図１１】図１の正誤記憶部に格納された正誤表の一例を示す。

【図１２】図１のデータ因果関係記憶部に格納されたデータ因果関係図の一例を示す図。

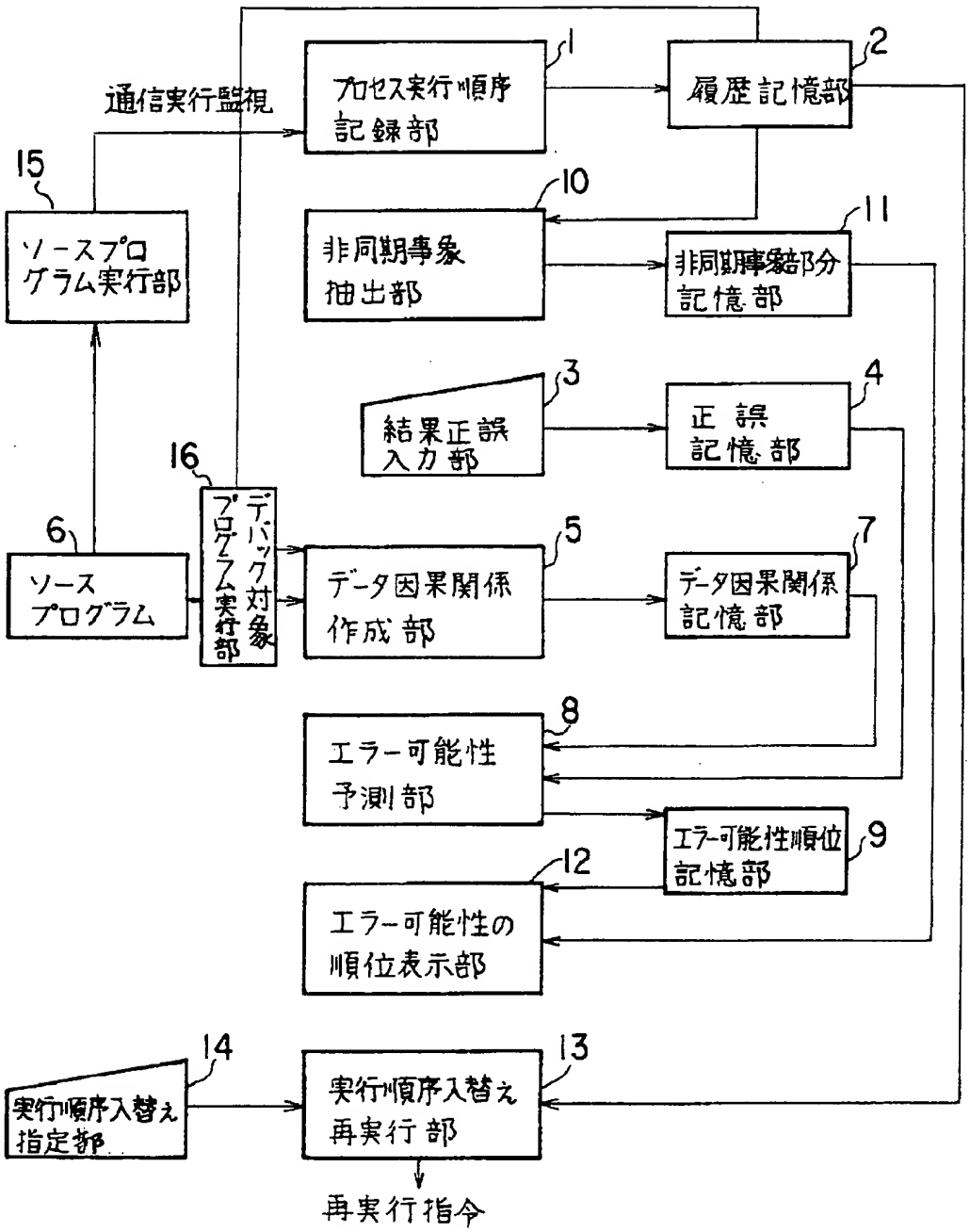
【図１３】図１のエラー順位記憶部に格納されたエラー順位表の一例を示す図。

【符号の説明】

１……プロセス実行順序記録部、２……履歴記憶部、３……正誤入力部、４……正誤記憶部、５……データ因果関係作成部、６……ソースプログラム、７……データ因果関係記憶部、８……エラー可能性予測部、９……エラー可能性順位記憶部、１０……非同期事象抽出部、１１……非同期事象部の記憶部、１２……表示部、１３……

実行順序入替え再実行部、１４……実行順序入替え指定部、１５……ソースプログラム実行部、１６……デバッグ対象プログラム実行部。

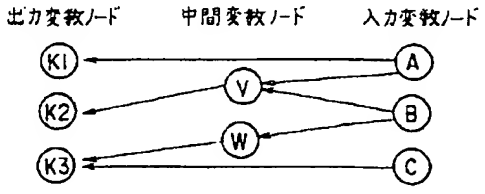
【図１】



【図１１】

結果	K1	K2	K3
正誤	誤	誤	正

【図１２】



【図２】

プロセス X

```
X1  wait(a,A)
X2  if(A > 3)
      K1 = A - 3 ・(1)
    else
      K1 = A ・・・・・(2)
X3  wait(a,A)
X4  wait(b,B)
X5  if(A > 1)
      V = A ・・・(3)
    else
      V = B ・・・(4)
X6  wait(b,B)
X7  w = B / 3
X8  wait(c,C)
X9  K2 = V * 3
X10 K3 = w + c
X11 プリント(K1,K2,K3)
```

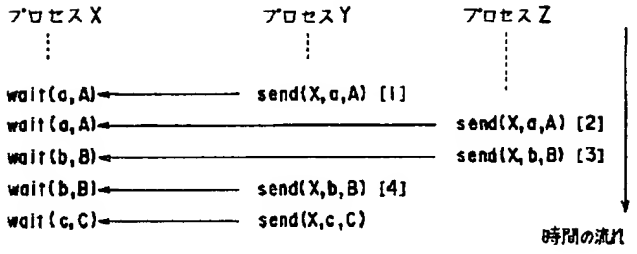
プロセス Y

```
Y1  send(X,a,A)
Y2  send(X,b,B)
Y3  send(X,c,C)
```

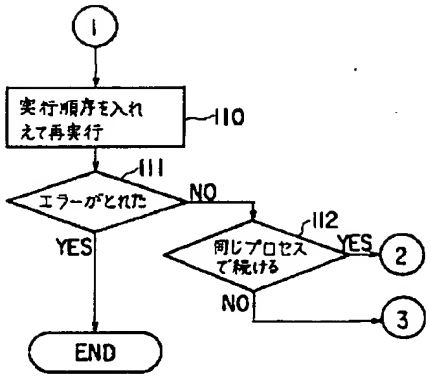
プロセス Z

```
Z1  send(X,a,A)
Z2  send(X,b,B)
```

【図３】



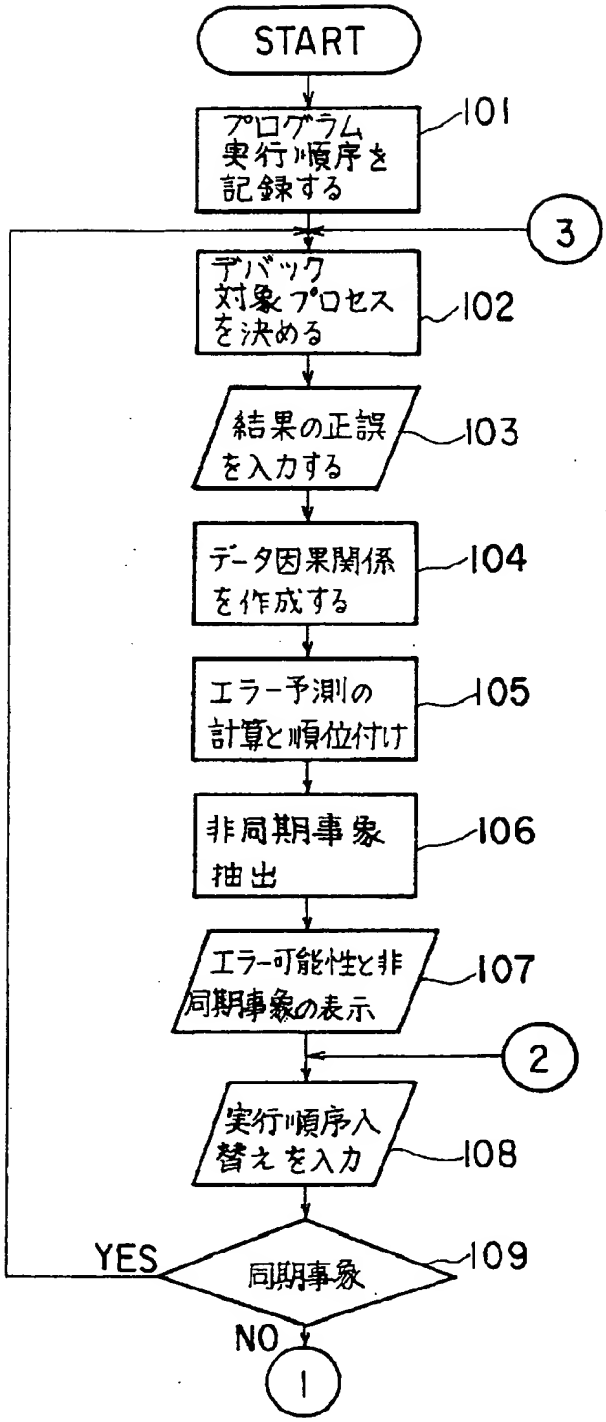
【図５】



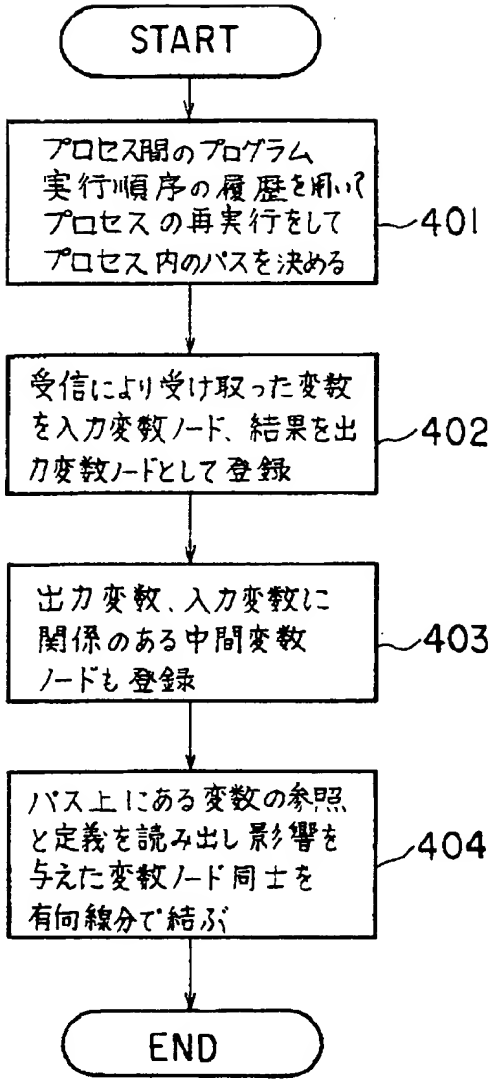
【図１０】

通信の種類	受信プロセス名	送信プロセス名	データ
send	X	Y	A
wait	X	Y	A
send	X	Z	A
wait	X	Z	A
send	X	Z	B
send	X	Y	B
wait	X	Z	B
wait	X	Y	B
send	X	Y	C
wait	X	Y	C

【図４】



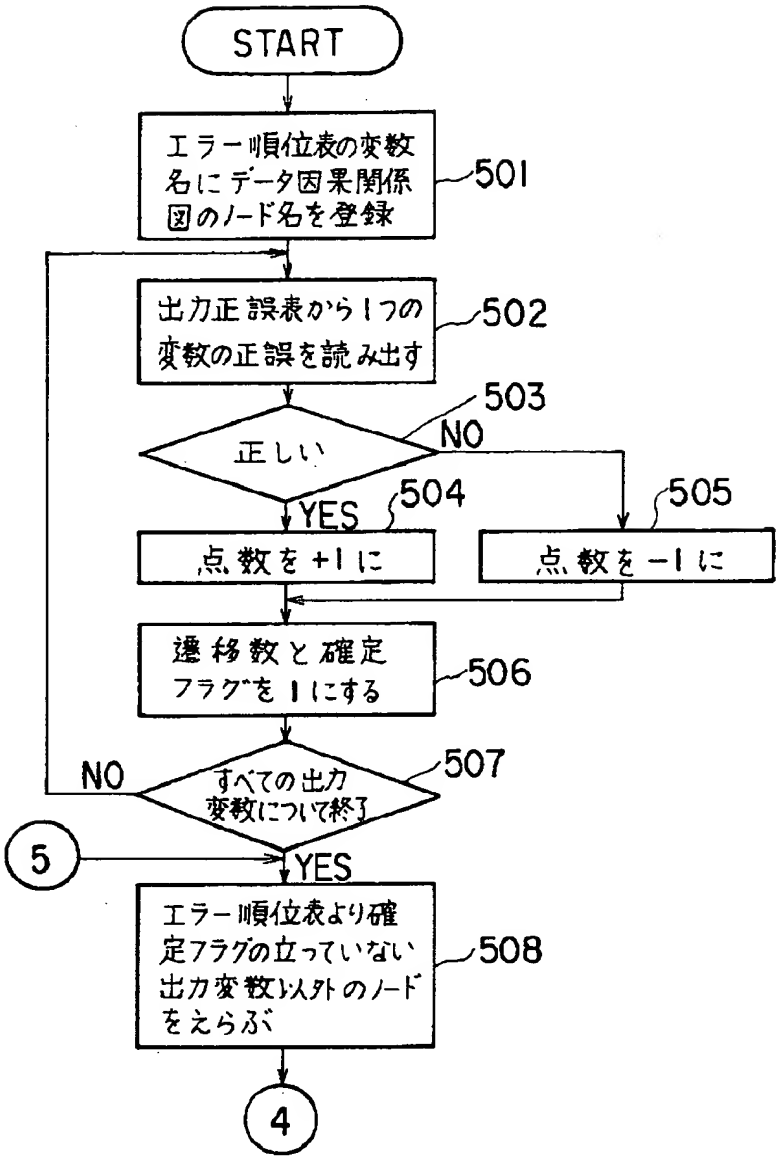
【図６】



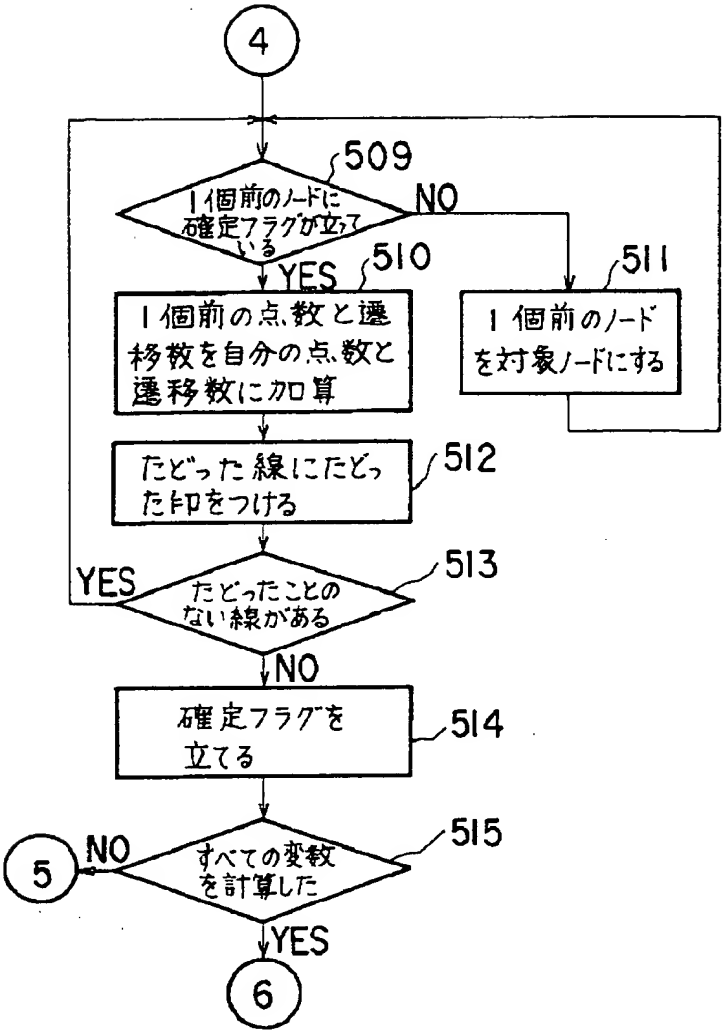
【図１３】

変数名	K1	K2	K3	V	W	A	B	C
点数	-1	-1	1	-1	1	-2	0	1
遷移数	1	1	1	1	1	2	2	1
確定フラグ	1	1	1	1	1	1	1	1
順位						1	2	3

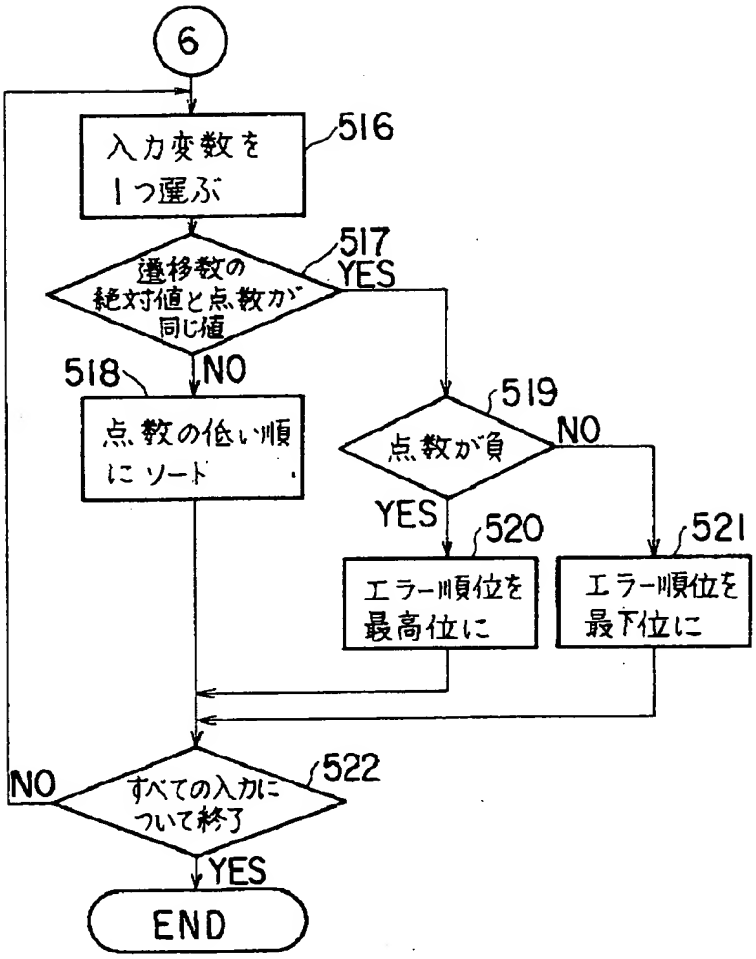
【図７】


START
エラー順位表の変数名にデータ因果関係図のノード名を登録
501
出力正誤表から1つの変数の正誤を読み出す
502
503
正しい
NO
YES
504
505
点数を+1に
点数を−1に
遷移数と確定フラグを1にする
506
NO
すべての出力変数について終了
507
5
YES
エラー順位表より確定フラグの立っていない出力変数以外のノードをえらぶ
508
4

【図８】


4
509
1個前のノードに確定フラグが立っている
NO
YES
510
511
1個前の点数と遷移数を自分の点数と遷移数に加算
1個前のノードを対象ノードにする
たどった線にたどった印をつける
512
YES
たどったことのない線がある
513
NO
確定フラグを立てる
514
5
NO
すべての変数を計算した
515
YES
6

【図９】


6
入力変数を
1つ選ぶ
516
遷移数の
絶対値と点数が
同じ値
517
YES
NO
518
点数の低い順
にソート
519
点数が負
NO
YES
520
エラー順位を
最高位に
521
エラー順位を
最下位に
すべての入力に
ついて終了
522
NO
YES
END